We claim:

1. A method for power control in a scalable pipelined array processor comprising the steps of:

controlling the execution of instructions in processing elements (PEs) and in a sequence processor (SP) by examining an S/P bit of each instruction;

masking off one or more PEs;

where a PE is masked off, allowing only PE communication DSU instructions to be decoded by the masked off PE, the masked off PE providing cluster switch control information in response to decoding a PE communication DSU instruction; and

maintaining all other execution units in masked off PEs in an inactive state to conserve power.

2. The method of claim 1 further comprising the steps of:

keeping the PEs off when an SP-only instruction is executing; and

gating the array clock off to the PEs.

3. A method for power control in a scalable pipelined array processor comprising the steps of:

controlling the execution of instructions in processing elements (PEs) and in a sequence processor (SP) by examining an S/P bit of each instruction;

masking off one or more PEs; and

maintaining all execution units in masked off PEs in an inactive state to conserve power.

4. The method of claim 3 further comprising the steps of:

keeping the PEs off when an SP-only instruction is executing; and

gating the array clock off to the PEs.

5. A reconfigurable register file system with port address latches and port usage control logic comprising:

an instruction register for storing an instruction specifying an operational requirement;

a reconfigurable register file comprising an odd register file having at least one data read port, and an even register file having at least one data read port;

an execution unit connected to said data read ports of the odd and even register files; and

port usage control logic connected to the instruction register and the reconfigurable register file to control the odd register file and the even register file port address input so that data read port lines change only as needed to support the operational requirement specified by the instruction.

6. The system of claim 5 further comprising:

at least one additional execution unit, and wherein said port usage control logic is further operable to control said at least one addition execution unit to transition from an active to an inactive state utilizing a clock gating mechanism.

7. The system of claim 5 further comprising:

at least one additional execution unit, and wherein said port usage control logic is further operable to control said at least one additional execution unit to transition from an active to an inactive status by forcing a no operation (NOP) condition to be determined in said at least one additional execution unit.

8. The system of claim 7 wherein the group bit field of an instruction in an instruction register is logically set to a control instruction setting which is treated as a NOP in said additional execution unit.

9. The system of claim 5 wherein the port usage control logic further comprises odd and even address control latches which are clocked every cycle in response to a clock input and which maintain their same state when required by the decoding of an instruction by recirculating their outputs back to odd and even input multiplexers, respectively.

10. The system of claim 9 wherein the port usage control logic further comprises gating circuitry for providing a control input to said odd and even input multiplexers whereby new data from a bit field of the instruction register is introduced into the odd and even register files for a first state and data at the outputs of the odd and even address control latches is recirculated for a second state.

11. The system of claim 5 wherein the port usage control logic further comprises:

a gating circuit connected to the reconfigurable register files and a clock input, the gating circuit being operable for gating the clock off so no change of state of the reconfigurable register files occurs for each cycle when change is not necessary and gating the clock on so new data is clocked into the reconfigurable system file for each cycle when change is desired.

12. The system of claim 5 further comprising an additional execution unit connected to further read ports of both the odd and even register files.

13. The system of claim 12 further comprising:

logical test circuitry for testing different bits in the instruction in the instruction register to determine whether one of said further read ports should not change state for the instruction.

14. A method for conserving power where during conditional execution of multi-cycle instructions execution conditions are determined during decode comprising the steps of:

determining for a multi-cycle instruction whether second and subsequent cycles of operation can be stopped based in part on an examination of one or more conditional execution flags during a first cycle; and

stopping operation at the end of the first cycle if the condition is determined to be such that no execution is to occur in the next cycle.

15. The method of claim 14 wherein said multi-cycle instructions are load/store instructions and load and store logic checks for their execution conditions during decode.

16. The method of claim 14 wherein said conditional execution comprises:

executing packed data type operations which can be conditional for each sub data type in a packed data operation.

17. The method of claim 16 wherein for those packed operations that take two or more cycles to execute, the method further comprises:

stopping execute cycles after the first execute cycle for those sub data operations which have condition flags indicating they are not to complete.

18. A method for conserving power where during conditional execution of multi-cycle instructions execution conditions are determined during the first execute cycle comprising the steps of:

determining for a multi-cycle instruction whether second and subsequent cycles of operation can be stopped based in part on an examination of one or more conditional execution flags during a first execute cycle; and

stopping operation at the end of the first cycle if the condition is determined to be such that no execution is to occur in the next cycle.

19. The method of claim 18 wherein said multi-cycle instructions are arithmetic instructions and the arithmetic instruction logic checks for their execution conditions during their first execution cycle.

20. The method of claim 19 wherein said arithmetic instructions are two cycle multiple instructions.

21. The method of claim 20 wherein said arithmetic instructions are two-cycle floating point instructions.

22. A method for conserving power in conjunction with register file indexing (RFI) comprising:

issuing an XV instruction with RFI enabled that reads a VLIW from a VIM register file for a PE or an SP;

using an indirect automatic increment of specified register operand read port addresses to access a new location in the VIM register files; and

reducing power utilization by not accessing the VIM for every subsequent XV instruction in a sequence of RFI-XV instructions.

23. The method of claim 22 wherein said step of reducing power further comprises the step of:

checking VIMOFFS and Vb fields for each subsequent XV instruction.

24. The method of claim 23 wherein if the VIMOFFS, contents of V0 and V1 VIM base address register (Vb), and Vb fields match those for the previous RFI-XV instruction, then reusing the VIM output with only register operands specified by the RFI operation updated.

25. A method for conserving power in conjunction with register file indexing (RFI) of a partitioned VIM unit comprising:

partitioning a VIM to associate a separate memory unit with each execution unit;

comparing a VIM address for each execution unit's separate memory unit by comparing VIMOFFS, contents of Vb, and Vb fields for each execution unit to a previous RFI-XV2 for each execution unit.

26. A method for conserving power by minimizing VIM accesses for a sequence of XV instructions which access the same VIM location, the method comprising the steps of:

issuing an XV instruction from the sequence that reads a VLIW from a VIM register file for a PE or an SP;

registering the VLIW in a VLIW storage register; and

reducing power utilization by not accessing the VIM for every subsequent XV instruction in the sequence of XV instruction and using the VLIW from the VLIW storage register.

27. The method of claim 26 wherein said step of reducing power further comprises the step of:

checking VIMOFFS, contents of V0 or V1 VIM base address register (Vb), and Vb fields for each subsequent XV instruction.

28. The method of claim 27 wherein if the VIMOFFS, contents of Vb, and Vb fields match those for the previous XV instruction, then reusing the VIM output.

29. The method of claim 27 wherein calculated VIM addresses for subsequent instructions in the sequence of XV instructions are compared with a stored version of the VIM address from the previous XV.

30. A method for conserving power in conjunction with a partitioned VIM unit comprising:

partitioning a VIM to associate a separate memory unit with each execution unit;

comparing a VIM address for each execution unit's separate memory unit by comparing VIMOFFS, contents of Vb, and Vb fields for each execution unit to a previous XV for each execution unit.

31. A method for power control in a reconfigurable register file system employing multiple execution units having associated register file ports for a reconfigurable register file, the method comprising the steps of:

independently controlling the associated register file ports such that the reconfigurable register file is logically treated as a n x m bit register file for a first execution unit and as a p x q bit register file for a second execution unit in a first cycle;

independently controlling the associated register file ports such that the reconfigurable register file is logically treated as a p x q bit register file for the first execution unit and as an n x m bit register file for the second execution unit in a second cycle; and

utilizing power saving port access control logic to independently control port usage for each execution unit for purposes of reducing system power utilization.

32. The method of claim 31 wherein n = 16, m = 64, p = 32 and q = 32.

33. A reconfigurable register file system employing multiple execution units having associated register file ports for a reconfigurable register file, the system comprising:

means for independently controlling the associated register file ports such that the reconfigurable register file is logically treated as a n x m bit register file for a first execution unit and as a p x q bit register file for a second execution unit in a first cycle;

means for independently controlling the associated register file ports such that the reconfigurable register file is logically treated as a p x q bit register file for the first execution unit and as an n x m bit register file for the second execution unit in a second cycle; and

port access control logic to independently control port usage for each execution unit for purposes of reducing system power utilization.

34. The system of claim 33 wherein $n = 16$, $m = 64$, $p = 32$ and $q = 32$.